PUC Goiás.

Goiânia, 8 de abril de 2017.

Aluno: Yan Américo Lobianchi da Cruz.

Disciplina: Estruturas de Dados I. Prof.: José Olímpio.

**Pesquisa: Lista Circular**

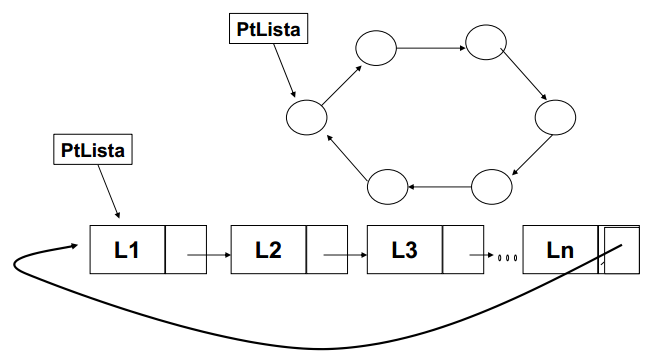
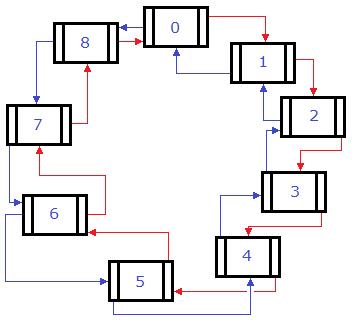
* Definição

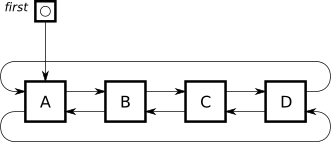
A lista circular é uma espécie de lista simplesmente ou duplamente encadeada, com uma característica adicional para o deslocamento na lista, "**ela não tem fim**".

Para tornar a lista interminável, o ponteiro **seguinte** do último elemento apontará para o **primeiro elemento** da lista, em vez do valor NULL, como vimos no caso das listas simplesmente e duplamente encadeadas.

Nas listas circulares, nunca chegaremos a uma posição a partir da qual não poderemos mais nos mover.

Chegando ao último elemento, o deslocamento vai recomeçar no primeiro elemento. Em suma, trata-se de uma rotação.





* Protótipo

O encadeamento circular em uma lista facilita as operações de inserção no fim e remoção no início, características de uma fila. A estrutura utilizada em uma lista circular é do mesmo tipo de uma lista dinâmica, simplesmente ou duplamente encadeada comum.

O elemento da lista conterá um campo **dado** e um ponteiro **seguinte**.   
O ponteiro **seguinte** será obrigatoriamente do mesmo tipo que o elemento, caso contrário ele não poderá apontar para o elemento.   
O ponteiro "**seguinte**" permitirá o acesso ao próximo elemento.

class ElementoLista {   
 char dado;   
 ElementoLista seguinte;   
};

Para controlar a lista, é melhor salvar certos elementos: o primeiro elemento, o último elemento, o número de elementos. Para tanto, outra estrutura vai ser utilizada (isso não é imprescindível, as variáveis podem ser utilizadas).   
Veja a composição:

class ListaDetectada {   
 ElementoLista início;   
 ElementoLista fim;   
 int tamanho;   
};

O ponteiro **início** integra o endereço do primeiro elemento da lista. O ponteiro de **fim** terá o endereço do último elemento da lista. A variável *tamanho* abrigará o número de elementos.

Seja qual for a posição na lista, os ponteiros **início** e **fim** sempre apontam, respectivamente, para o primeiro e último elemento. O campo *tamanho* abrigará o número de elementos da lista, independentemente da operação efetuada na lista.

Na Lista Circular Duplamente Encadeada só muda que cada nó possui dois ponteiros: um para o elemento anterior e outro para o próximo elemento (ant e prox). O anterior do primeiro é o último e o próximo do último é o primeiro.

* Operações
* Inicialização

Protótipo da função:

void inicia(Lista lista);

Dita operação deve ser realizada antes de qualquer outra operação na lista.   
Ela inicializa o ponteiro **início** e o ponteiro de **fim** sempre através do ponteiro NULL, e o tamanho, com o valor 0. 

* Inserção de um elemento na lista

Veja o algoritmo de inserção e de backup dos elementos: Declaração de elementos a serem inseridos:

* Alocação da memória para o novo elemento;
* Preenchimento do conteúdo do campo de dados;
* A atualização dos ponteiros com alvo no primeiro e no último elemento, se necessário;
* Caso especial: em uma lista com um único elemento, o primeiro também é o último;
* Atualização do tamanho da lista.
* Inserção em uma lista vazia

Protótipo da função:

int ins\_lista\_circ\_vazia(Lista lista, char dado);

A função volta em -1 com falha, se não ela retorna 0.   
  
Passos :

* Alocação da memória para o novo elemento;
* Preenchimento do campo de dados do novo elemento;
* O ponteiro **seguinte** do novo elemento apontará para ele mesmo (***é a etapa que torna a lista circular***);
* Os ponteiros **início** e de **fim** apontarão para o novo elemento;
* O tamanho é atualizado.
* Inserção em uma lista não vazia

Protótipo da função:

int ins\_lista\_circ(Lista lista, Elemento em andamento, char dado);

A função exibe-se novamebne -1 em caso de falha, se não ela retorna 0.   
A inserção será efetuada no final da lista.   
  
Passos:

* Alocação da memória para o novo elemento;
* Preenchimento do campo de dados do novo elemento;
* O ponteiro **seguinte** do novo elemento aponta para o endereço do primeiro elemento (***manter a lista circular***);
* O ponteiro **início** não muda;
* O ponteiro **fim** aponta para o novo elemento;
* O tamanho é incrementado com uma unidade.

* Remoção de um elemento da lista

Algoritmo de remoção de um elemento da lista:

* Utilização de um ponteiro temporário para salvar o endereço de elementos a serem removidos;
* O elemento a ser suprimido se encontra depois do elemento em andamento;
* Fazer o ponteiro **seguinte** do elemento em andamento apontar para o endereço do ponteiro **seguinte** do elemento a ser removido;
* Liberar a memória ocupada pelo elemento excluído;
* Atualizar o tamanho da lista.

Para remover um elemento da lista, há várias situações:

1. Remover do início da lista;
2. Remover da lista;
3. Remover o último elemento da lista.

* Remoção no início da lista

Protótipo da função:

**int** remov\_lista\_circ**(**Lista lista**);**

Etapas:

* O ponteiro de remoçãoconterá o endereço do primeiro elemento;
* O ponteiro **início** apontará para o segundo elemento;
* O ponteiro **seguinte** do último elemento apontará para o primeiro elemento (**que era o segundo, antes da remoção**.);
* O tamanho da lista será decrementado de um elemento.
* Remoção em uma lista comum apenas um elemento

Protótipo da função:

int remov\_lista\_circ\_única (Lista lista);

Etapas:

* O ponteiro **remov\_elem** conterá o endereço do elemento (a lista contém apenas um elemento);
* O ponteiro **início** apontará para **NULL**;
* O ponteiro **fim** apontará para **NULL**;
* O tamanho da lista será decrementado de um elemento.
* Exibir a lista

Para exibir a lista inteira você deverá se posicionar no início da lista (isso é possível com o ponteiro **início**). Em seguida, ao usar o ponteiro **seguinte** de cada elemento, a lista é percorrida do primeiro ao último elemento.

Em comparação com as listas simples e duplamente encadeadas, onde a condição de parada é dada pelo ponteiro **seguinte** do último elemento, que equivale a **NULL** para a lista circular, não há ponto de parada, a menos que você escolha um.

Variantes de exibição: exibição da lista (do primeiro para o último elemento) e exibição da lista, sem condição de parada (ao infinito).

* Destruição da lista

Na destruição da lista, a remoção no início da lista enquanto o tamanho fosse superior a 1; em seguida, a remoção em uma lista com apenas um elemento. Pode ser feita de outras maneiras.

**Bibliografia**

<http://br.ccm.net/faq/10226-listas-circulares-ring-buffer>

<http://www.pognao.com.br/2012/09/lista-com-encadeamento-circular-em-c.html>

<http://www.inf.ufrgs.br/~galante/wiki/lib/exe/fetch.php?id=inf01203&cache=cache&media=inf01203-aula18-listacircular.pdf>

<http://www.san.uri.br/~ober/arquivos/disciplinas/estruturaII_SI/listas_duplamente_encadeadas.ppt>